Engenharia de Software Moderna

Cap. 4 - Modelos

Prof. Marco Tulio Valente

https://engsoftmoderna.info

Motivação

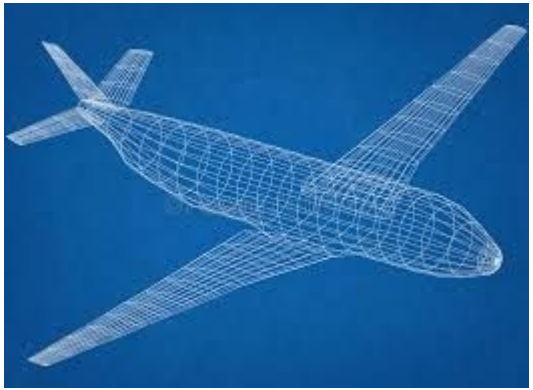
- Existe uma lacuna entre os seguintes mundos:
  - Requisitos: **o que** o sistema faz (abstração mais alta)
  - Código: **como** o sistema faz isso (abstração mais baixa)

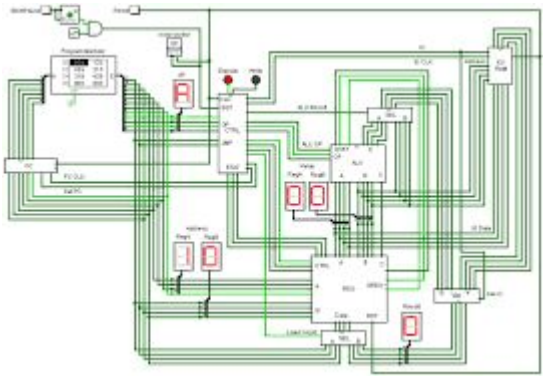
Modelos de Software

- Objetivo: preencher essa "lacuna"
- Via uma notação com um nível de abstração intermediário
- Documentar uma solução para o problema definido pelos requisitos

Comuns em outras Engenharias

- Natural que fossem propostos também para software



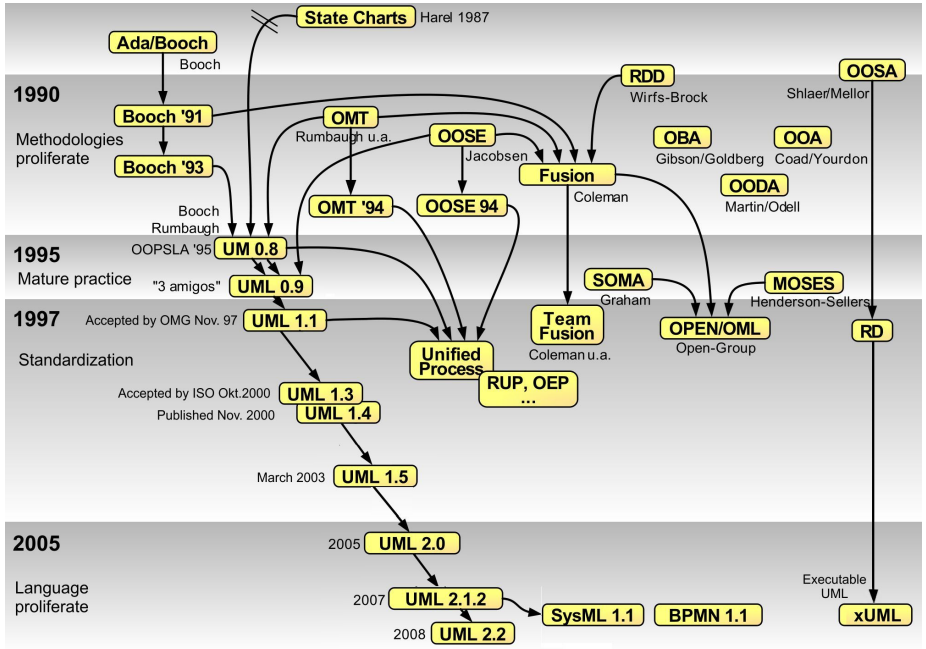


Modelos de Software

- Infelizmente, não são tão efetivos e largamente usados, como em outras engenharias
- Modelos de software podem ser:
  - Formais: menos comuns; não serão estudados aqui
  - Gráficos: UML é a notação mais comum

UML: Unified Modelling Language

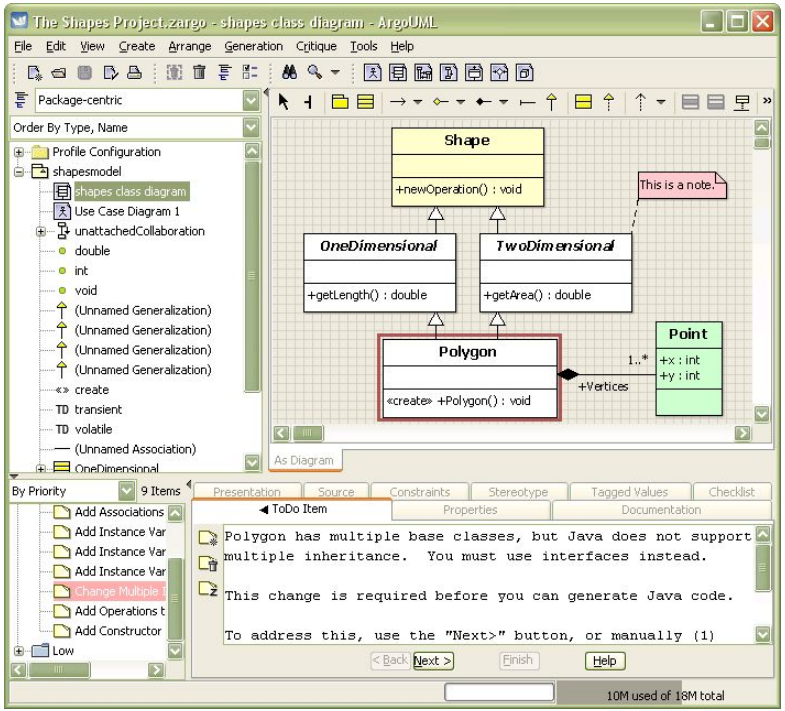
- Proposta em 1995, para fundir outras notações



Fonte: Wikipédia

Ferramentas CASE (Computer-Aided Software Engineering)

- Equivalente a ferramentas CAD, mas para Eng. Software



Quais os principais usos de UML?

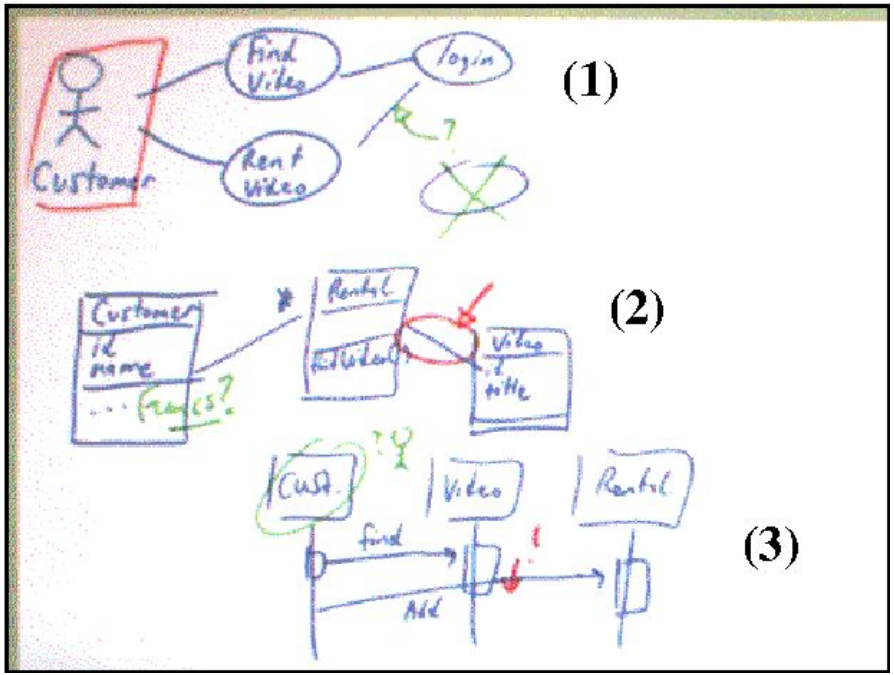
1. Como blueprint (planta detalhada)
2. Como sketches (esboços, rascunhos)

Neste curso, vamos estudar o uso de UML como sketches

UML como Sketch

- Uso mais comum de UML com métodos ágeis
- Uso mais informal e leve da notação
- Objetivo **não** é ter um modelo completo
- UML é usada para:
  - Conversar sobre uma parte do código ou do projeto
  - Documentar uma parte do código ou do projeto

UML como Sketch



Q. Chen, J. Grundy, J. Hosking: SUMLOW: early design-stage sketching of UML diagrams on an E-whiteboard. Software Practice and Experience, 2008

Sketches UML são úteis em

Engenharia Avante e em Engenharia Reversa

Engenharia Avante (“Forward”)

- Modelo é usado para discutir alternativas de projeto
- Antes de qualquer linha de código ser implementada

Engenharia Reversa

- Modelo é usado para explicar um código que já existe
- Contextos de manutenção e evolução de software

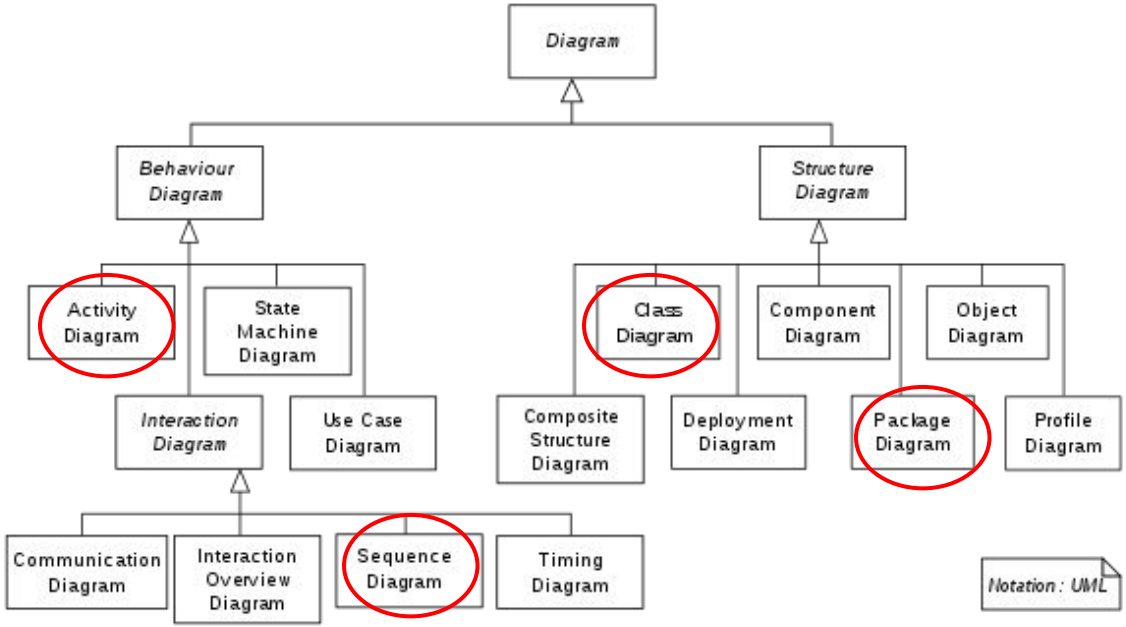
Diagramas UML

Diagramas UML

- Diagramas Estáticos: modelam a estrutura do código
- Diagramas Dinâmicos: modelam a execução do código (o comportamento do sistema)

Diagramas UML

Em vermelho, os diagramas que vamos estudar



Versão de UML que iremos usar

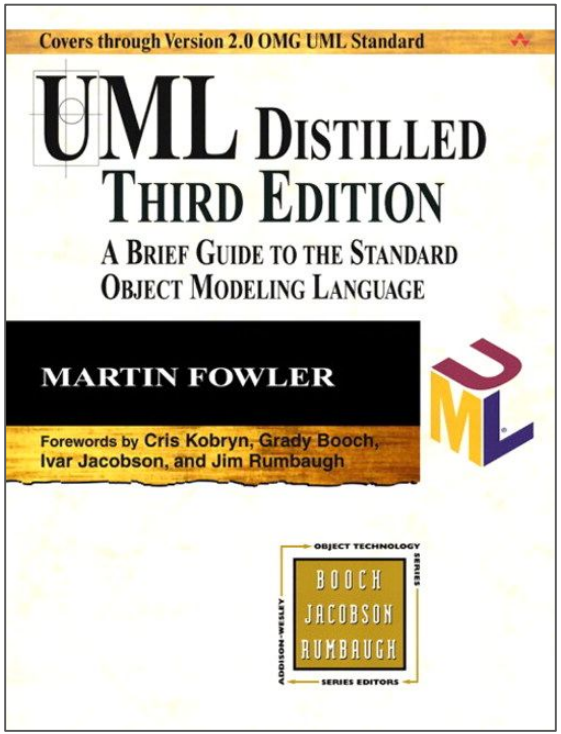
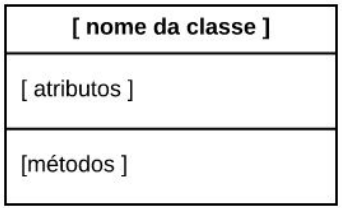
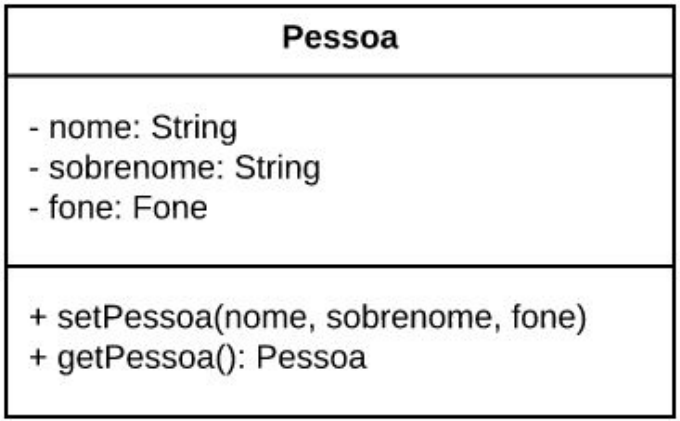


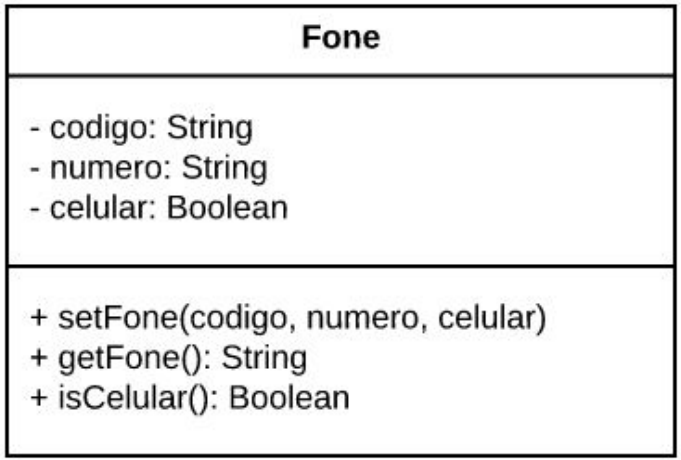
Diagrama de Classes

Formato genérico



Exemplo com duas classes

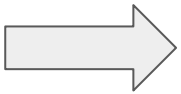


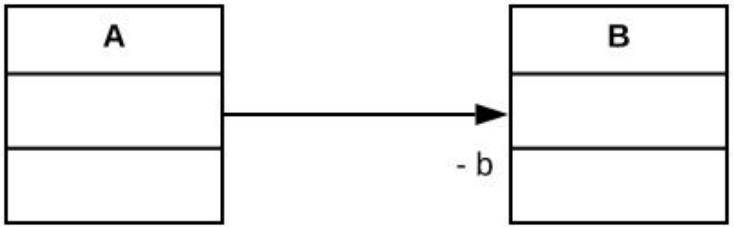


- : private
+: public

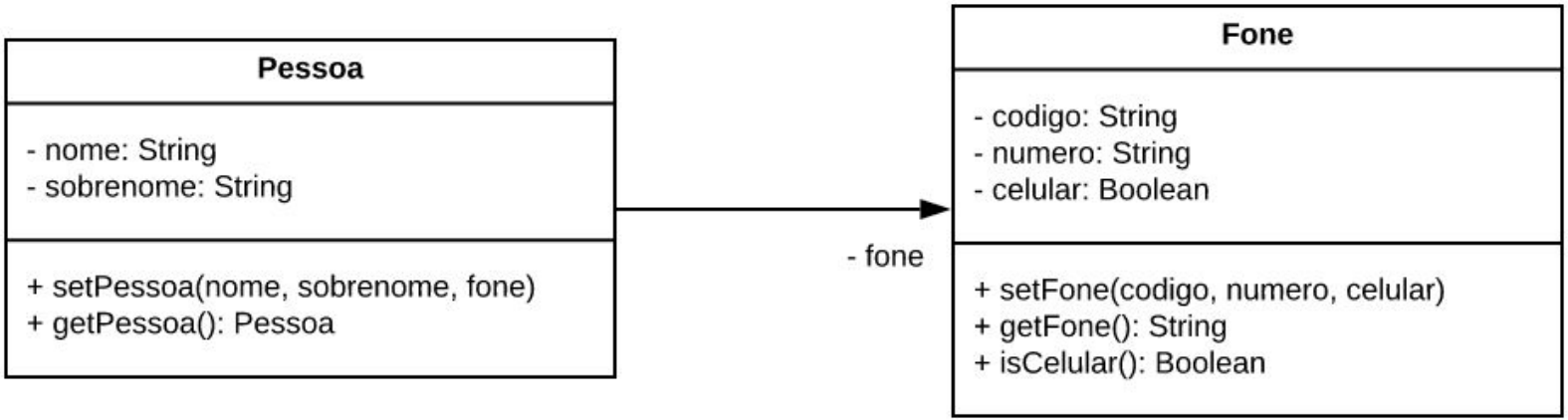
Associações

```
class A {
    ...
    private B b;
    ...
}


class B {
    ...
}
```

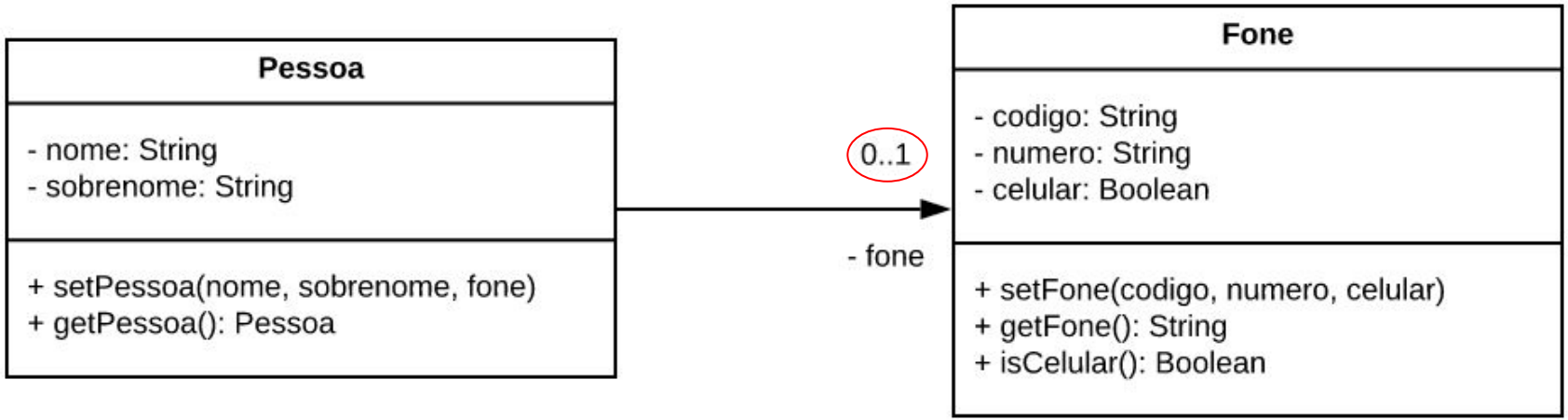




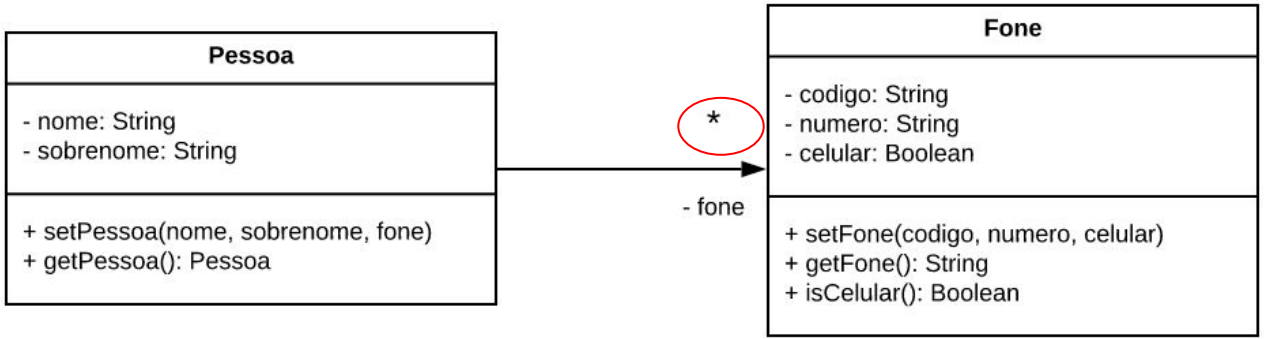
Associações



Multiplicidade (exemplo 1)

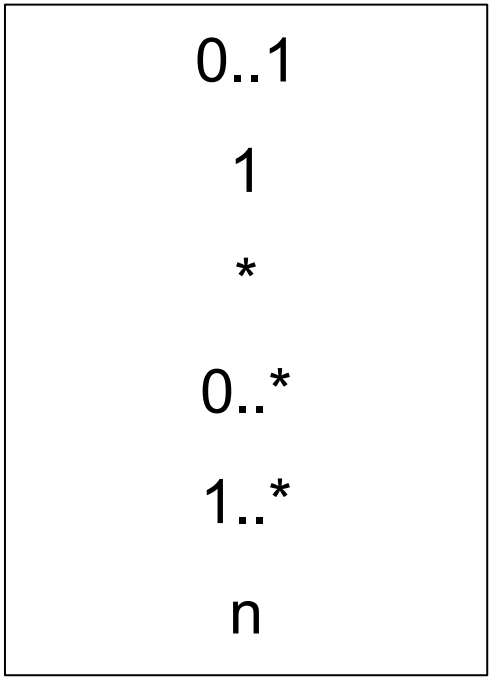


Multiplicidade (exemplo 2)

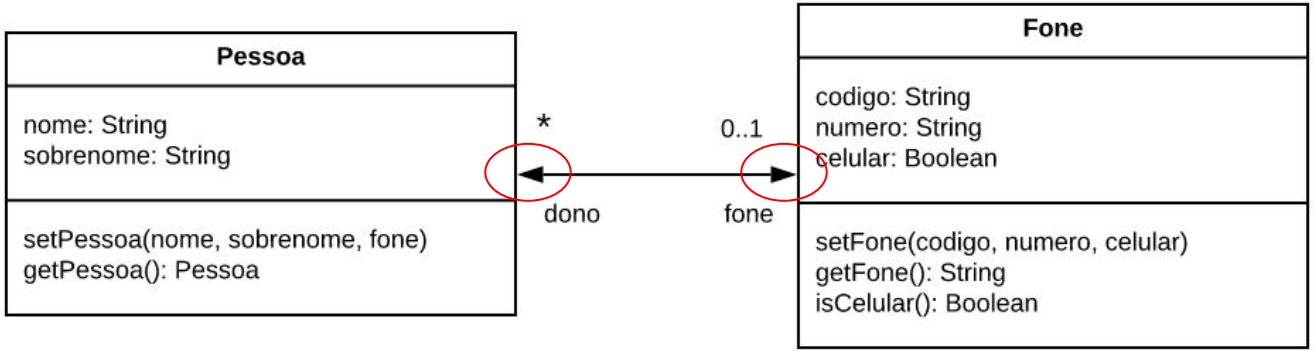



```
class Pessoa {
    private Fone[] fone;
    ...
}
class Fone {
    ...
}
```

Principais multiplicidades

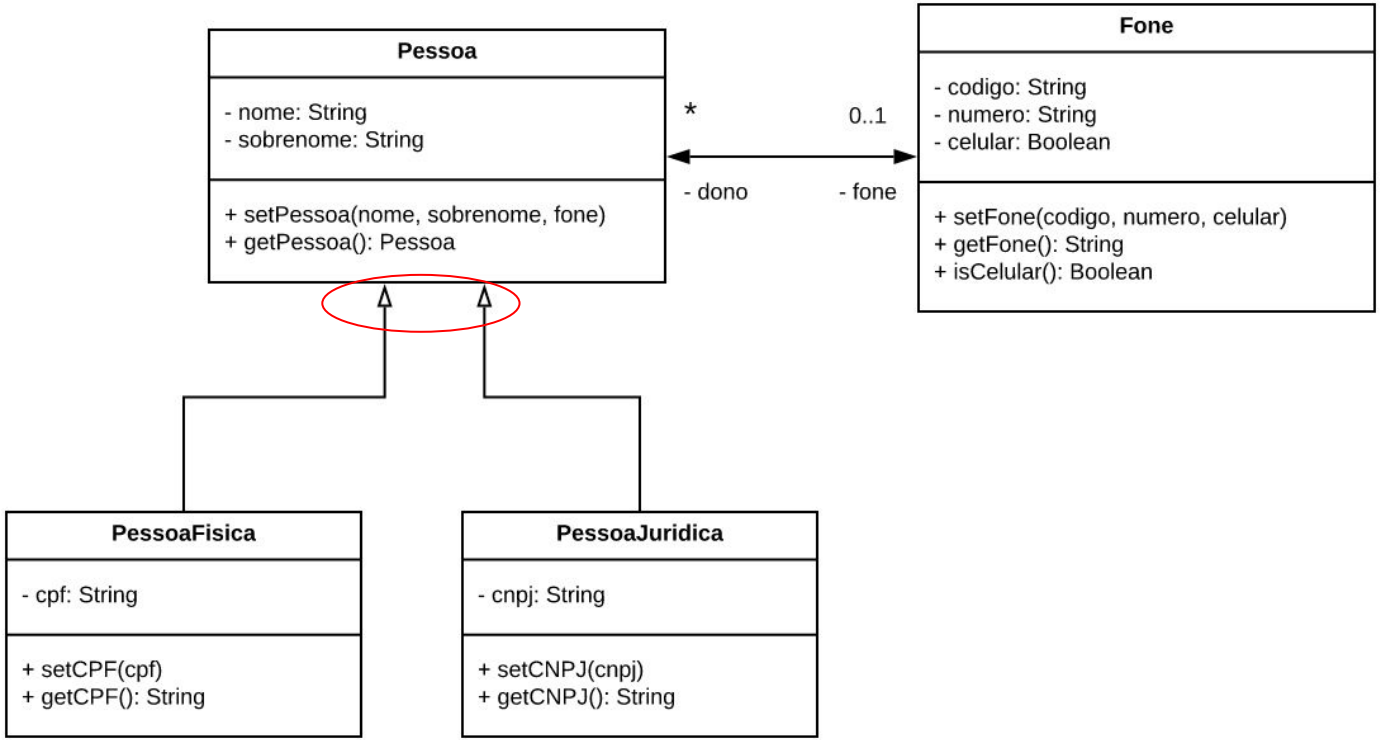


Associação bidirecional




```
class Pessoa {
    ...
    private Fone fone;
    ...
}

class Fone {
    ...
    private Pessoa[] dono;
    ...
}
```

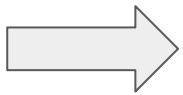
Herança

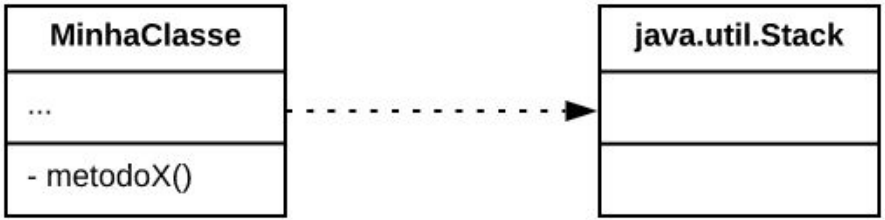

Pessoa
- nome: String
- sobrenome: String
+ setPessoa(nome, sobrenome, fone)
+ getPessoa(): Pessoa
*
0..1
- dono
- fone
Fone
- codigo: String
- numero: String
- celular: Boolean
+ setFone(codigo, numero, celular)
+ getFone(): String
+ isCelular(): Boolean
PessoaFisica
- cpf: String
+ setCPF(cpf)
+ getCPF(): String
PessoaJuridica
- cnpj: String
+ setCNPJ(cnpj)
+ getCNPJ(): String

Dependências (setas tracejadas)

Relacionamento entre duas classes, mas que não é devido a associação ou herança

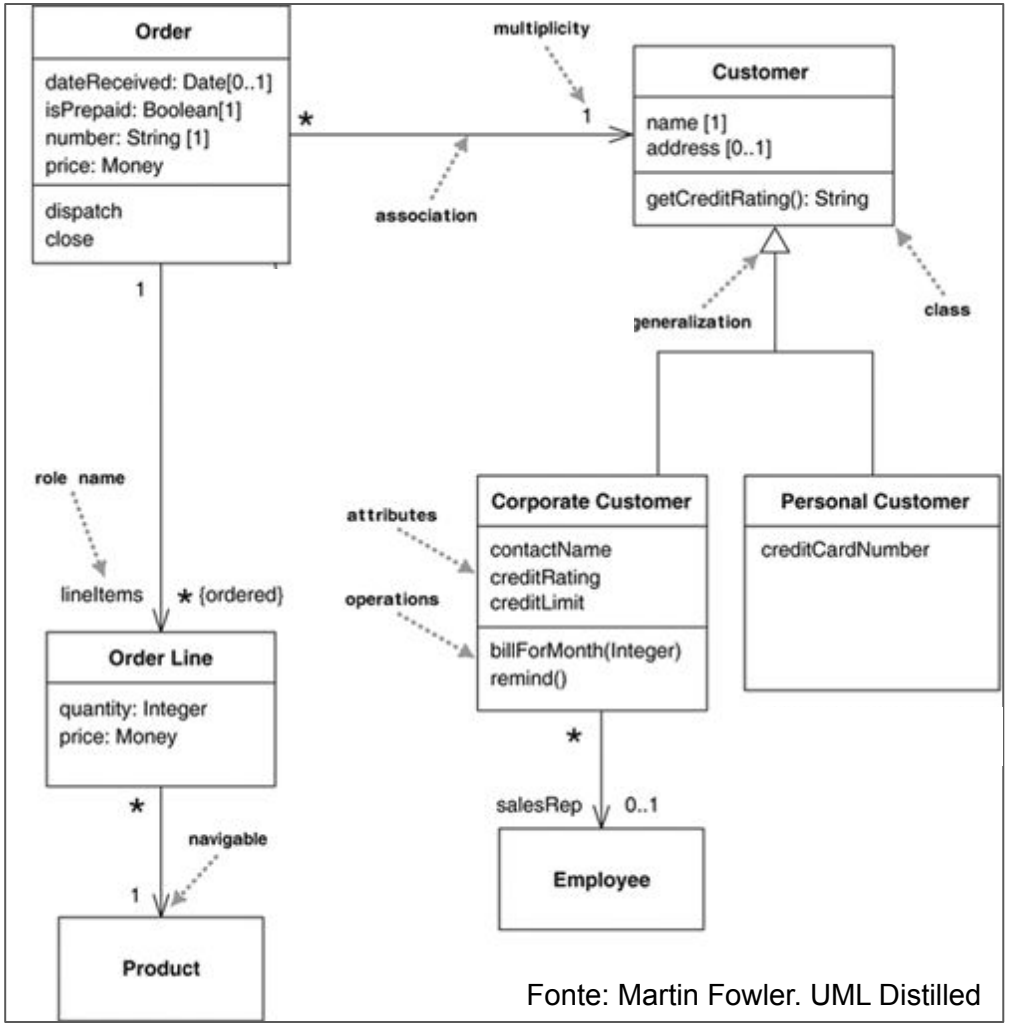
```
import java.util.Stack;

class MinhaClasse {
    ...
    private void metodoX() {
      Stack stack = new Stack();
      ...
    } ...
}
```





Dependências não possuem informação sobre multiplicidade

Exercícios sobre Diagrama de Classes



Fonte: Martin Fowler. UML Distilled

1. Estude e procure entender o seguinte diagrama de classes

2. Modele os cenários descritos a seguir usando Diagramas de Classe UML. Veja que as classes são grafadas em uma fonte diferente.

- `ContaBancaria` possui exatamente um `Cliente`. Mas um `Cliente` pode ter várias `ContaBancaria`, com navegabilidade nos dois sentidos.
- `ContaPoupanca` e `ContaSalario` são subclasses de ContaBancaria.
- No código de `ContaBancaria` declara-se uma variável local do tipo `BancoDados`.
- Um `ItemPedido` se refere a um único `Produto` (sem navegabilidade). Um `Produto` pode ter vários `ItemPedido` (com navegabilidade).
- A classe `Aluno` possui atributos nome, matricula, curso (todos privados); e métodos getCurso() e cancelaMatricula(), ambos públicos.

3. Crie diagramas de classes para os seguintes trechos de código:

```
class HelloFrame {
   public static void main(String[] args) {
     JFrame frame = new JFrame("Hello!");
     frame.setVisible(true);
   }
}
```

```
class HelloFrame extends JFrame {
   public HelloFrame() {
     super("Hello!");
   }
   public static void main(String[] args) {
     HelloFrame frame = new HelloFrame();
     frame.setVisible(true);
   }
}
```

Diagrama de Pacotes

Diagrama de Pacotes

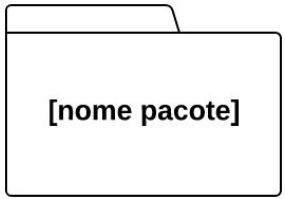


Diagrama de Pacotes

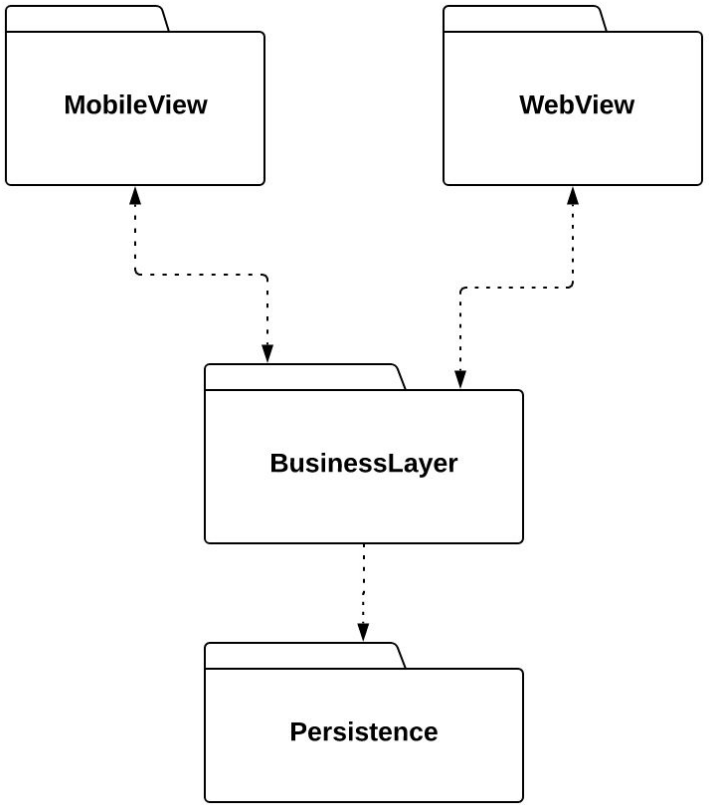


Diagrama de Sequência

Diagramas de Sequência

- São diagramas comportamentais ou dinâmicos
- Modelam:
  - Objetos de um sistema
  - Métodos que eles executam

Diagrama de Sequência (exemplo 1)

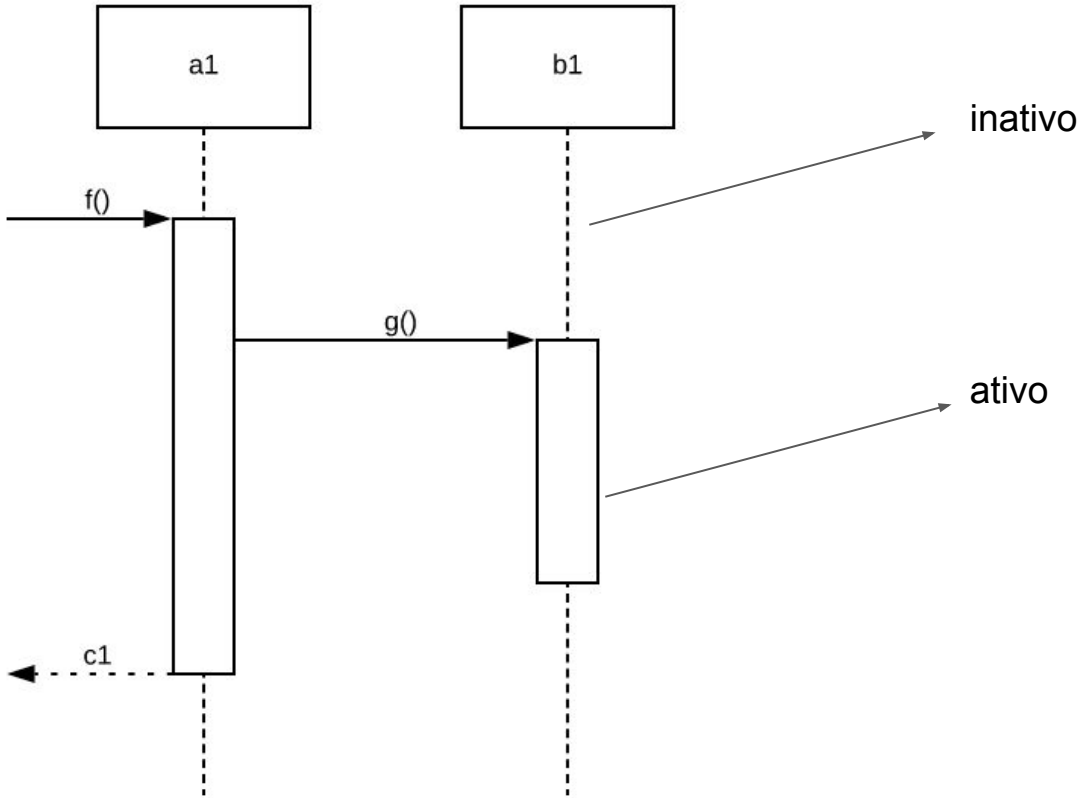
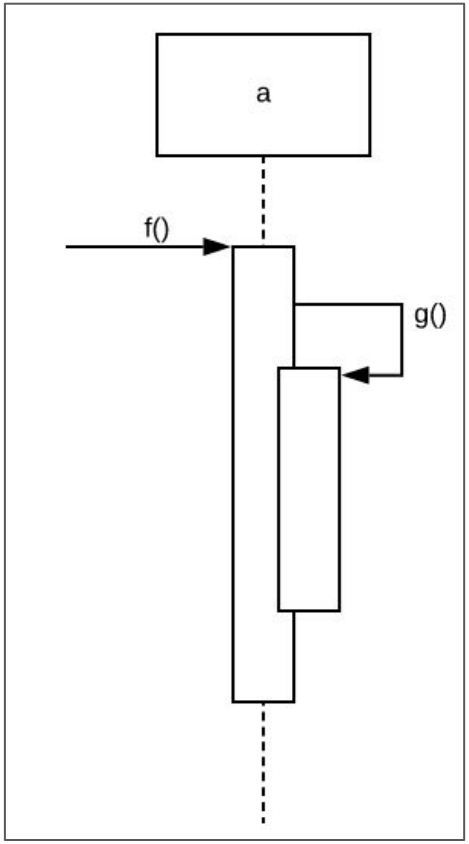


Diagrama de Sequência (exemplo 2)

```
class A {

  void g() {
    ...
  }

  void f() {
    ...
    g();
    ...
  }

  main() {
    A a = new A();
    a.f();
  }
}
```



Setas de retorno de métodos

- Muitas vezes, são omitidas:
  - Porque o retorno não é importante
  - Porque o método não retorna valor (void)
- Fowler:
  - Some people use returns for all calls, but I prefer to use them only where they add information; otherwise, they simply clutter things.

Diagrama de Sequência (exemplo 3)

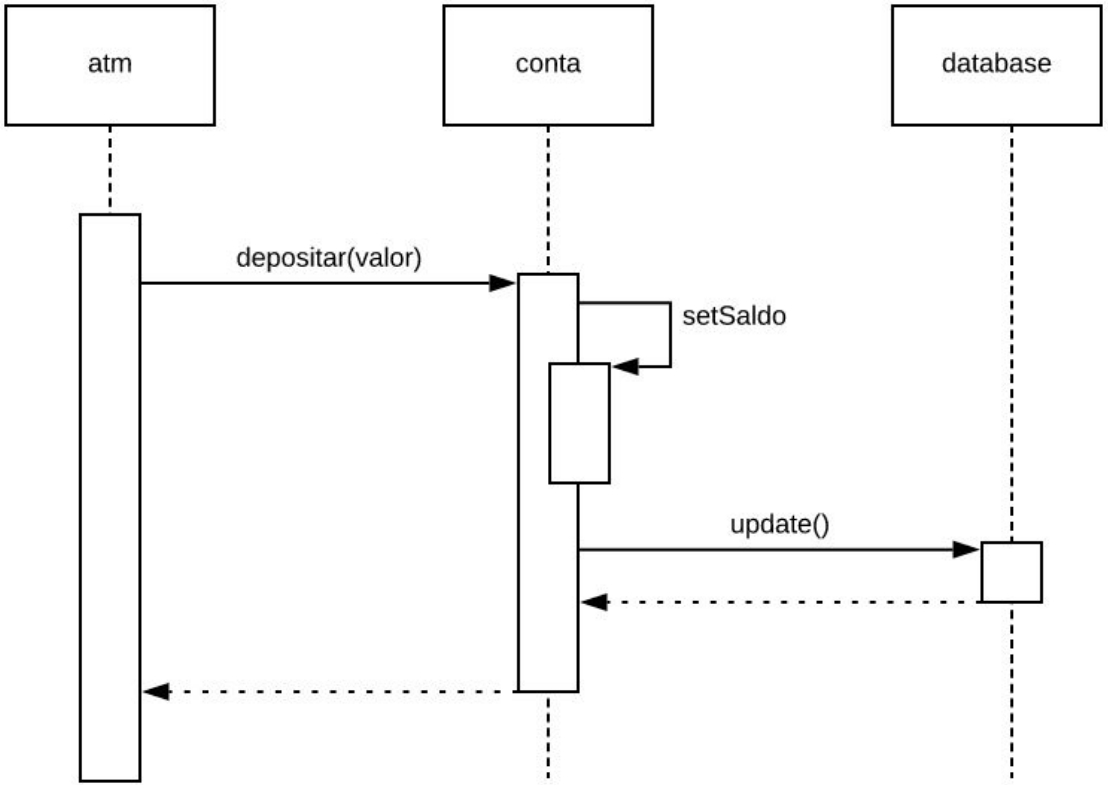
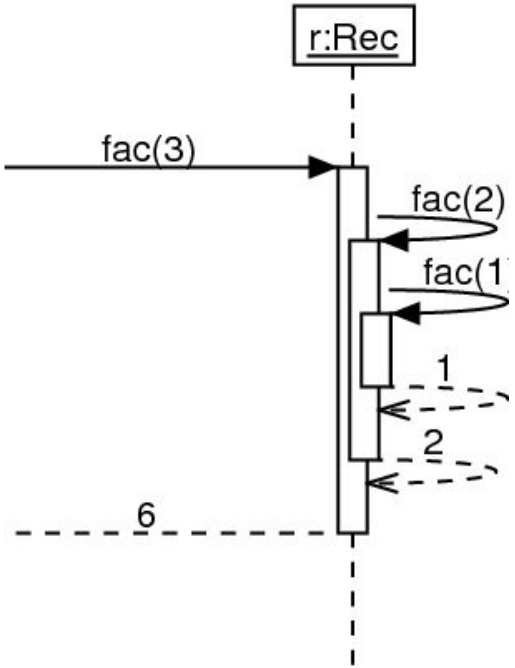


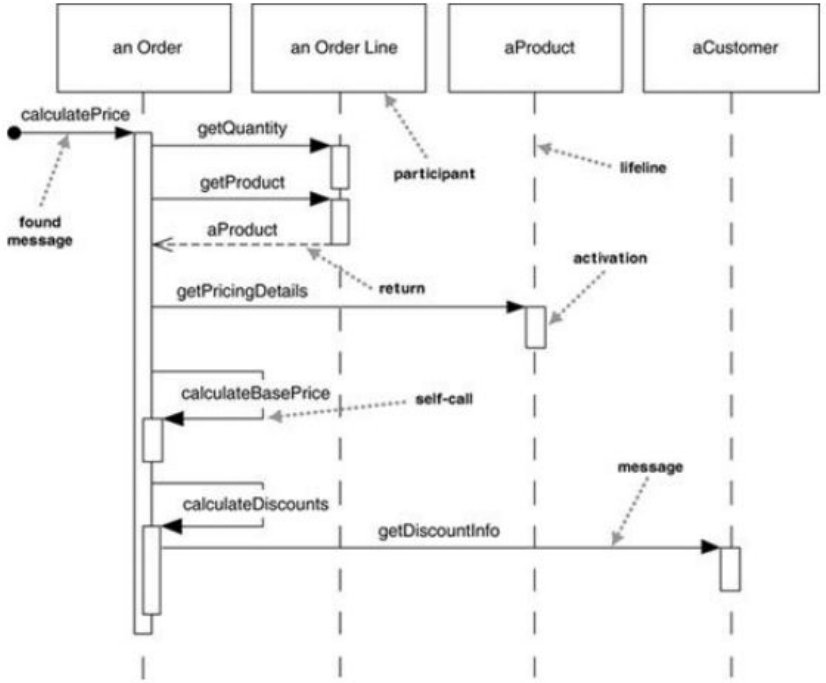
Diagrama de Sequência (exemplo 4)



Evidentemente, não é um uso interessante de diagramas de sequência

Exercícios

Esse diagrama de sequência deveria representar o processamento necessário para calcular o valor total de um Pedido (Order), o qual possui diversos Itens (Lines), cada um referindo-se a um Produto e com uma quantidade. Por que ele, no entanto, não faz isso corretamente?



Versão Correta do Diagrama Anterior

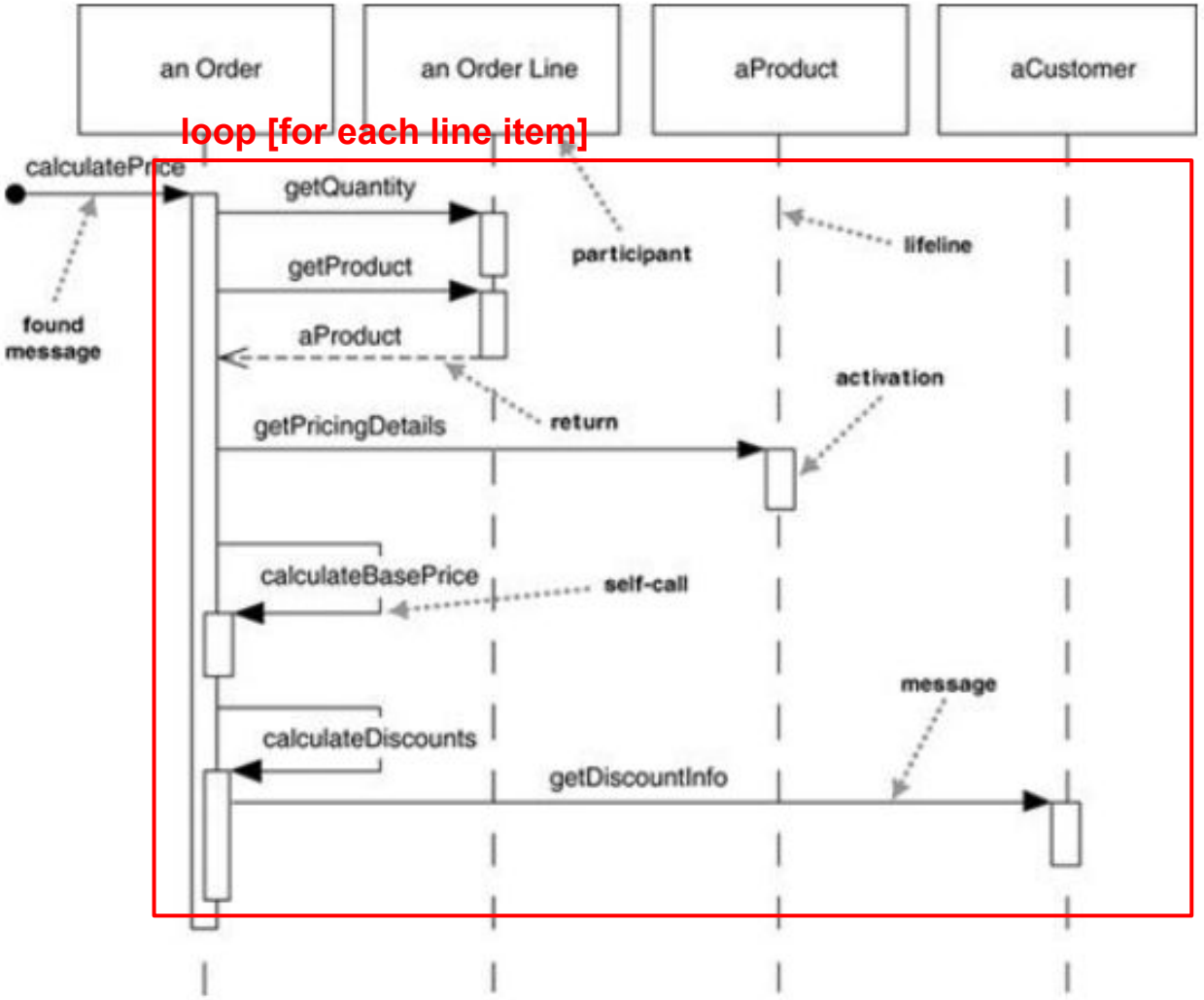
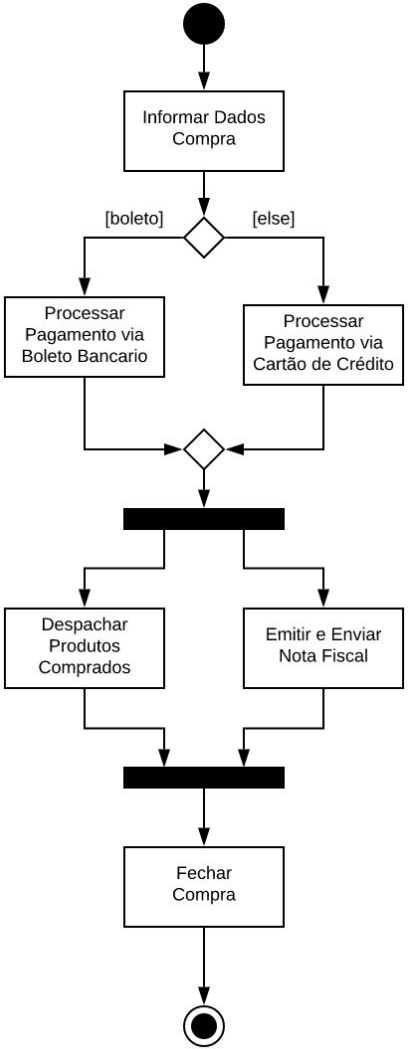


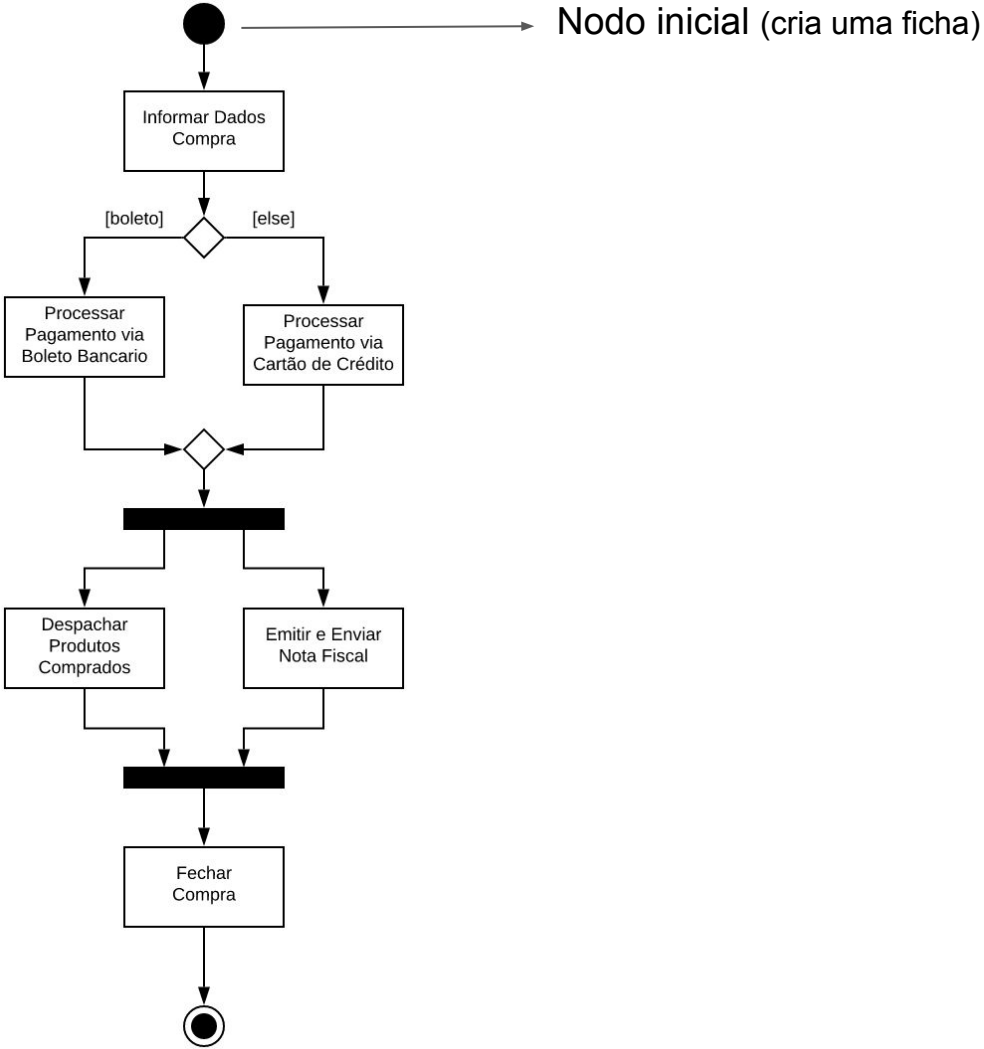
Diagrama de Atividades

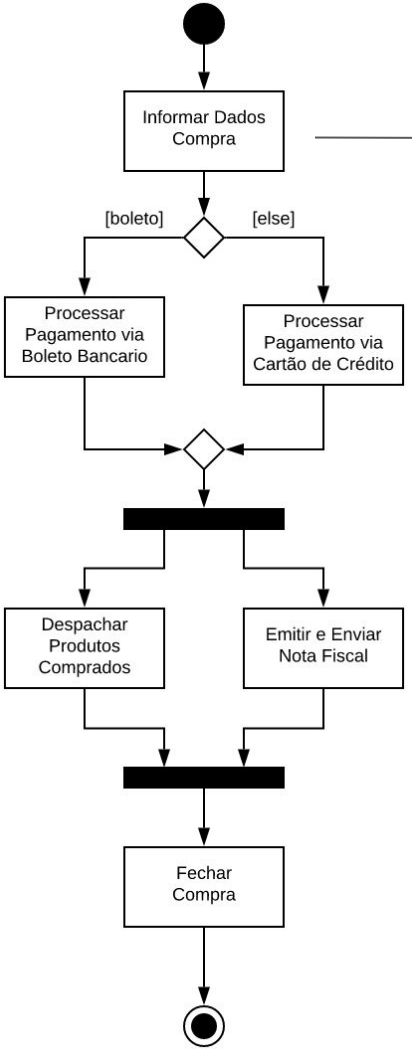
Diagramas de Atividades

- Também são diagramas comportamentais ou dinâmicos
- Modelam em alto nível um processo ou fluxo de negócio

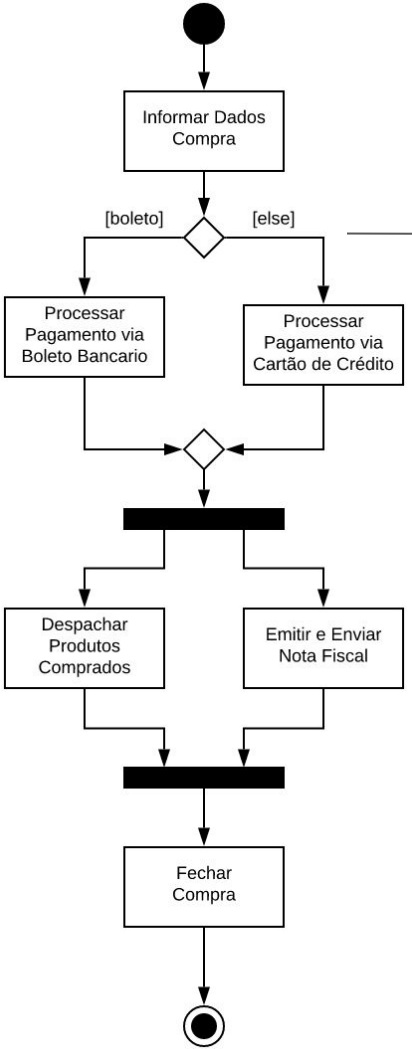


Imagine que existe uma ficha (token) que caminha pelos nodos do diagrama de atividades.

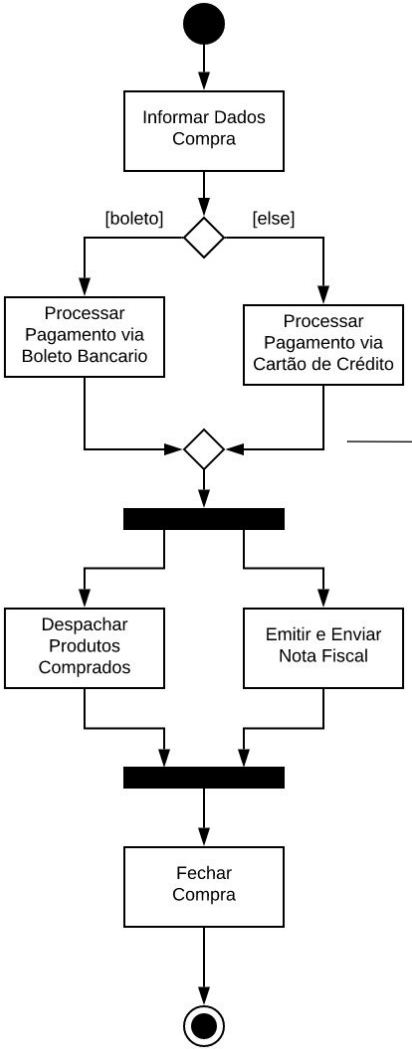

Nodo inicial (cria uma ficha)
Informar Dados Compra
[boleto]
[else]
Processar Pagamento via Boleto Bancario
Processar Pagamento via Cartão de Crédito
Despachar Produtos Comprados
Emitir e Enviar Nota Fiscal
Fechar Compra



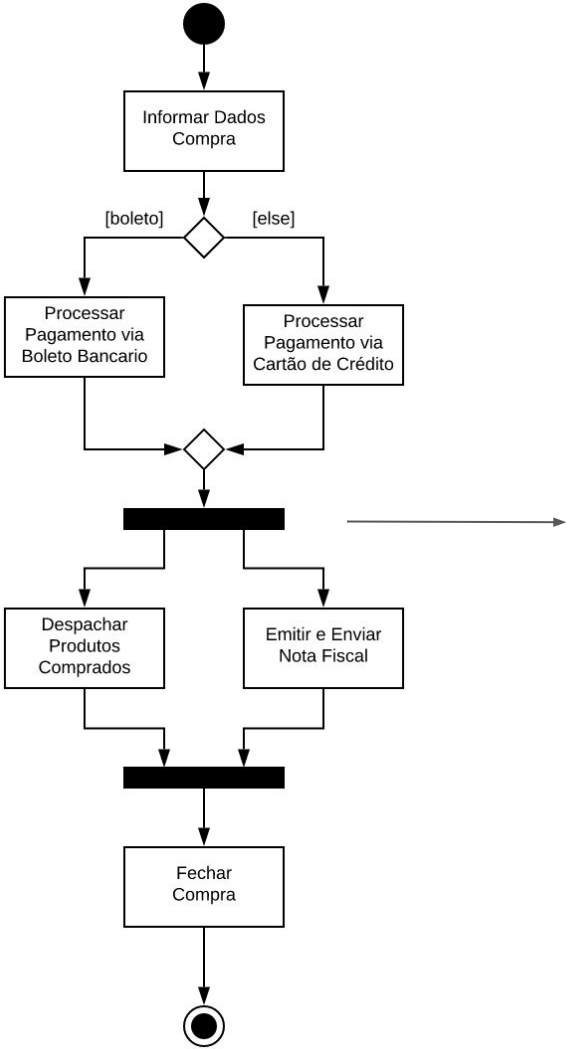
Ação (repassa ficha do fluxo de entrada para saída)



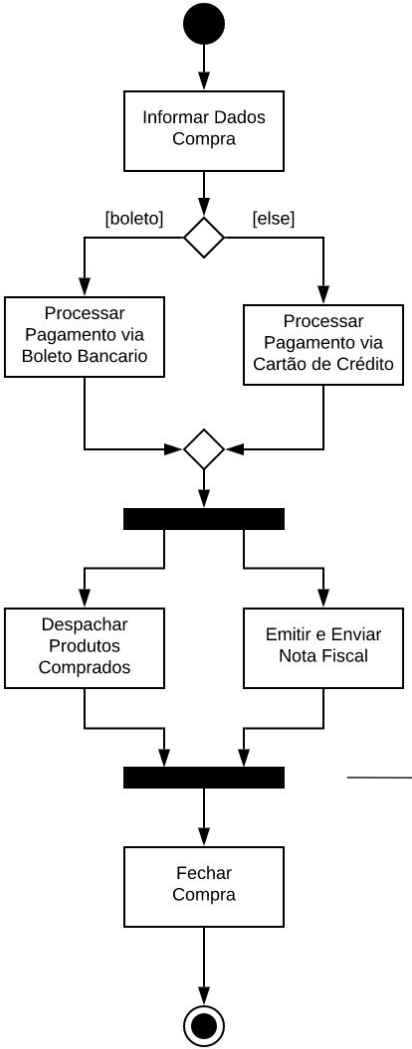
Decisão (decide para qual fluxo de saída repassará a ficha)



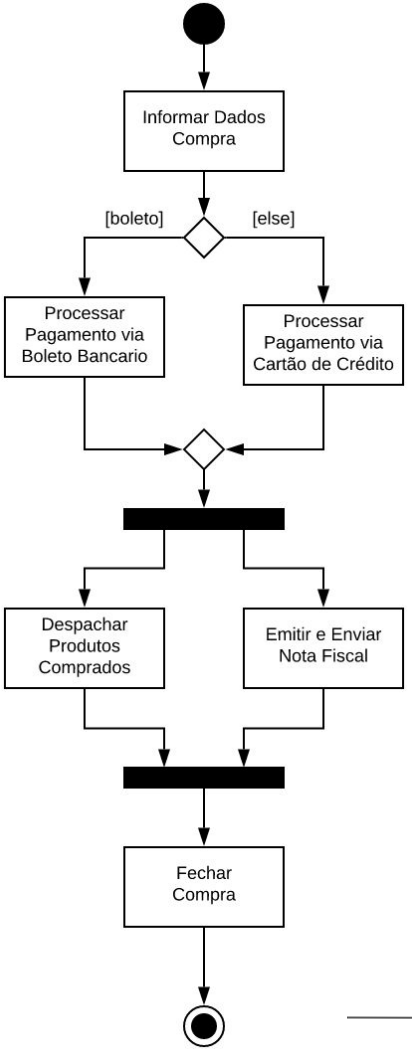
Merge (quando ficha chega em uma das entradas, repassa para saída)



Fork (multiplicadores de fichas)



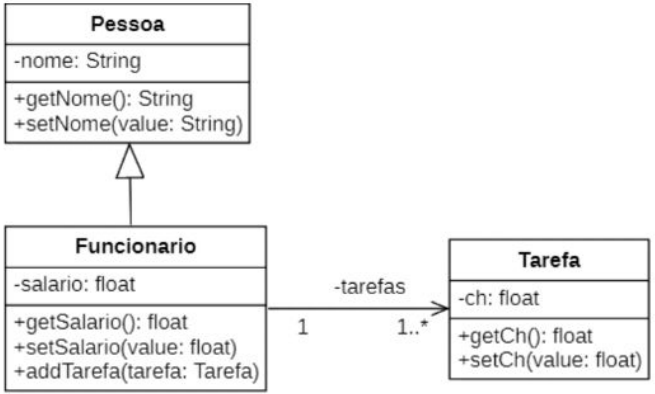
Join (sorvedouros de fichas)



Nodo final (encerra execução)

Exercícios

1. (Poscomp 2023) Seja o seguinte diagrama de classes. Marque a alternativa correta.



(A) classe “Pessoa” se associa com a classe “Funcionario”, que por sua vez tem uma relação de generalização com a classe “Tarefa”.

(B) “Pessoa” herda da classe “Funcionario”, que tem uma relação de associação com “Tarefa”.

(C) A associação com navegabilidade da classe “Funcionario” para a classe “Tarefa” gera no código um atributo “lista de objetos” da classe “Tarefa” na classe “Funcionario”.

(D) A classe “Tarefa” faz parte da classe “Funcionario”, constituindo uma relação de agregação.

(E) O método “addTarefa(tarefa: Tarefa)” pode ser invocado a partir de uma instância da classe “Pessoa”, através de polimorfismo.

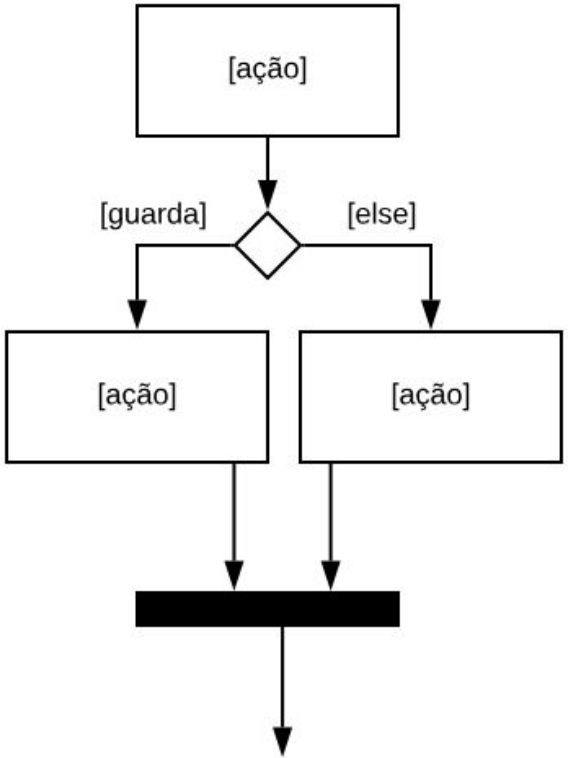
2. Modele em UML usando um Diagrama de Classes.

```
class Computador {
   ...
   private List<Teclado> teclados;
   ...
}
```

```
class Teclado {
  ...
}
```

Observação: Teclado não possui uma referência de volta para Computador. Porém, no nosso sistema, sabemos que qualquer Teclado está sempre ligado a exatamente um Computador.

3. Qual é o erro do seguinte diagrama de atividades? Refaça o diagrama de forma a refletir corretamente a intenção do projetista.



Fim